

图1-16 不同数据类型的存储（注：图片来源为 Behrouz Forouzan[6]，2008年12月）

位 (bit) 是存储在计算机中的最小数据单位，它是0或1。位代表设备的某一状态，例如：用1表示开关合上，0表示断开。为了表示数据的不同类型，使用位模式，它是一个序列，是0和1组和。通常长度位8的位模式称为1个字节 (byte)。

接下来我们来看文本 (Text) 是如何存储的。在任何语言中，文本都是由一些符号组成。位模式可以表示任何一个符号。需要多少位来表示一个符号取决于该语言使用的符号的数量，如表1-2所示。

表1-2 符号数量和位模式长度

符号数目	位模式的长度
2	1
4	2
8	3
16	4
256	8
65536	16

不同的位模式集合被设计用于表示文本符号，每个集合被称为代码表，表示符号的过程称为编码。美国国家标准协会 (American National Standards Institute, 记为 ANSI) 发布了美国信息交换标准码 (American Standard Code for Information Interchange, 记为 ASCII) 的代码表。该代码使用一串 7位二进制数表示每个符号，可以定义  $2^7=128$  种不同的符号，包括英语中常用的 26 个大写字母，26 个小写字母，9 个字符，以及标点符号等。如图 1-17 所示，展示了四个大写字母对应的二进制位模式表示。

**B**      **Y**      **T**      **E**  
 |            |            |            |  
**1000010    1011001    1010100    1000101**

图1-17 大写字母的二进制位模式表示

编程语言，如果注释也写成英文，使用ASCII中包含的字符就够了。ASCII为控制字符保留了前32个代码，这些代码最初的目的不是为了携带可打印信息，而是为了控制使用ASCII的设备（如打印机）。例如，十进制字符10代表“换行”功能（使打印机推进纸张），字符27代表“转义”键，经常出现在普通[键盘]的左上角。代码127（全部七位开启），另一个特殊字符，相当于“删除”或“擦除”。需要掌握的是图1-18中划线标识的代码，10换行，13回车，48-57表示数字0-9，65-90表示大写字母，97-122表示小写字母。如果记不住确切的对应十进制字符，记住大写字母在小写字母之前也可以。

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000	NUL	(null)	32	20 040	&#32;	Space		64	40 100	&#64;	Ø	96	60 140	&#96;	`		
1	1 001	SOH	(start of heading)	33	21 041	&#33;	!	65	41 101	&#65;	A	97	61 141	&#97;	a			
2	2 002	STX	(start of text)	34	22 042	&#34;	"	66	42 102	&#66;	B	98	62 142	&#98;	b			
3	3 003	ETX	(end of text)	35	23 043	&#35;	#	67	43 103	&#67;	C	99	63 143	&#99;	c			
4	4 004	EOT	(end of transmission)	36	24 044	&#36;	\$	68	44 104	&#68;	D	100	64 144	&#100;	d			
5	5 005	ENQ	(enquiry)	37	25 045	&#37;	%	69	45 105	&#69;	E	101	65 145	&#101;	e			
6	6 006	ACK	(acknowledge)	38	26 046	&#38;	&	70	46 106	&#70;	F	102	66 146	&#102;	f			
7	7 007	BEL	(bell)	39	27 047	&#39;	'	71	47 107	&#71;	G	103	67 147	&#103;	g			
8	8 010	BS	(backspace)	40	28 050	&#40;	(	72	48 110	&#72;	H	104	68 150	&#104;	h			
9	9 011	TAB	(horizontal tab)	41	29 051	&#41;	)	73	49 111	&#73;	I	105	69 151	&#105;	i			
10	A 012	LF	(NL line feed, new line)	42	2A 052	&#42;	*	74	4A 112	&#74;	J	106	6A 152	&#106;	j			
11	B 013	VT	(vertical tab)	43	2B 053	&#43;	+	75	4B 113	&#75;	K	107	6B 153	&#107;	k			
12	C 014	FF	(NP form feed, new page)	44	2C 054	&#44;	,	76	4C 114	&#76;	L	108	6C 154	&#108;	l			
13	D 015	CR	(carriage return)	45	2D 055	&#45;	-	77	4D 115	&#77;	M	109	6D 155	&#109;	m			
14	E 016	SO	(shift out)	46	2E 056	&#46;	.	78	4E 116	&#78;	N	110	6E 156	&#110;	n			
15	F 017	SI	(shift in)	47	2F 057	&#47;	/	79	4F 117	&#79;	O	111	6F 157	&#111;	o			
16	10 020	DLE	(data link escape)	48	30 060	&#48;	0	80	50 120	&#80;	P	112	70 160	&#112;	p			
17	11 021	DC1	(device control 1)	49	31 061	&#49;	1	81	51 121	&#81;	Q	113	71 161	&#113;	q			
18	12 022	DC2	(device control 2)	50	32 062	&#50;	2	82	52 122	&#82;	R	114	72 162	&#114;	r			
19	13 023	DC3	(device control 3)	51	33 063	&#51;	3	83	53 123	&#83;	S	115	73 163	&#115;	s			
20	14 024	DC4	(device control 4)	52	34 064	&#52;	4	84	54 124	&#84;	T	116	74 164	&#116;	t			
21	15 025	NAK	(negative acknowledge)	53	35 065	&#53;	5	85	55 125	&#85;	U	117	75 165	&#117;	u			
22	16 026	SYN	(synchronous idle)	54	36 066	&#54;	6	86	56 126	&#86;	V	118	76 166	&#118;	v			
23	17 027	ETB	(end of trans. block)	55	37 067	&#55;	7	87	57 127	&#87;	W	119	77 167	&#119;	w			
24	18 030	CAN	(cancel)	56	38 070	&#56;	8	88	58 130	&#88;	X	120	78 170	&#120;	x			
25	19 031	EM	(end of medium)	57	39 071	&#57;	9	89	59 131	&#89;	Y	121	79 171	&#121;	y			
26	1A 032	SUB	(substitute)	58	3A 072	&#58;	:	90	5A 132	&#90;	Z	122	7A 172	&#122;	z			
27	1B 033	ESC	(escape)	59	3B 073	&#59;	:	91	5B 133	&#91;	[	123	7B 173	&#123;	{			
28	1C 034	FS	(file separator)	60	3C 074	&#60;	<	92	5C 134	&#92;	\	124	7C 174	&#124;				
29	1D 035	GS	(group separator)	61	3D 075	&#61;	=	93	5D 135	&#93;	]	125	7D 175	&#125;	}			
30	1E 036	RS	(record separator)	62	3E 076	&#62;	>	94	5E 136	&#94;	^	126	7E 176	&#126;	~			
31	1F 037	US	(unit separator)	63	3F 077	&#63;	?	95	5F 137	&#95;	_	127	7F 177	&#127;	DEL			

Source: [www.LookupTables.com](http://www.LookupTables.com)

图1-18 ASCII代码表（注：图片来源为lookuptables.com。增加了两条横线和三个矩形框，用来突出重点部分）

如果有Python环境，可以在命令行中，输出ASCII表内容。

```

1 In[1]: import string
2 In[2]: string.printable
3
4 Out[2]:
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'( )*+,-.:/;=>?
@[\\\]^_`{|}~ \t\n\r\x0b\x0c'

```

第 2 行的 `string.printable` 是调用了 `string` 类中的 `printable` 函数。如果想知道 `string` 中包含哪些函数，可以 `dir`。

```

1 In[3]: dir(string)
2 Out[3]:
3 ['Formatter',
4  'Template',
5  '_ChainMap',
6  '_TemplateMetaclass',
7  '__all__',
8  '__builtins__',
9  '__cached__',
10  '__doc__',
11  '__file__',
12  '__loader__',
13  '__name__',
14  '__package__',
15  '__spec__',
16  '_re',
17  '_sentinel_dict',
18  '_string',
19  'ascii_letters',
20  'ascii_lowercase',
21  'ascii_uppercase',
22  'capwords',
23  'digits',
24  'hexdigits',
25  'octdigits',
26  'printable',
27  'punctuation',
28  'whitespace']

```

ASCII 有结构特点。数字 0-9 以二进制的方式表示，其数值前缀为 0011。小写字母和大写字母在位模式上只有一位的差别，这就把大小写转换简化为一个范围测试（以避免转换不是字母的字符）和一个单一的比特操作。快速的大小写转换很重要，因为它经常被用于大小写搜索算法中。

例子：将任何 ASCII 字母变成小写字母。

在 ASCII 中，大写字母和小写字母的区别在于位 00100000 (十六进制表示是 20h) 的值，该位在小写字母中被打开。如果 "打开" 这个位，大写字母就会变成小写字母。(如果该字母已经是小写字母，20h位已经打开；打开它没有任何区别)。"打开" 20h 位，被称为 ORing-in 位，因为使用的是位布尔 OR 操作符 (bitwise Boolean OR operator, [http://teaching.idallen.com/cst8214/08w/notes/bit\\_operations.txt](http://teaching.idallen.com/cst8214/08w/notes/bit_operations.txt)) 。

```
1   ...
2       01000001 = 41h = ASCII upper-case letter 'A'
3
4 OR      00100000 = 20h <-- this is the bit we want turned on
5
6 -----
7
8 EQUALS 01100001 = 61h = ASCII lower-case letter 'a'
9   ...
10
11 uppara = ord('A')
12 lowera = uppara | 0x20      # bitwise OR with 20h
13 print(chr(lowera))
14
15 lowera = uppara | (1<<5)
16 print(chr(lowera))
17
18
19 lowera = ord('a')
20 uppara = lowera & ~0x20      # bitwise AND with 10111111
21 print(chr(uppara))
22
23 uppara = lowera & ~(1<<5)
24 print(chr(uppara))
25
26 #a
27 #a
28 #A
29 #A
```

## 第2章 编程语法

在开启计算思维实践之前，需要先掌握一门编程语法。本书以 Python 语言为主，因为它容易上手、功能丰富，是一种极受欢迎的解释型高级编程语言，目前被广泛使用。它支持结构化编程和面向对象编程，拥有动态类型系统和垃圾回收功能，能够自动管理内存使用，并且其本身拥有标准库。Python 语法简洁，使用空格缩进划分代码块。在 1991 年 2 月，吉多·范罗苏姆（荷兰语：Guido van Rossum，1956 年 1 月 31 日－）发布 Python，截至 2022 年 6 月，python.org 推出的最新版本是 3.10.4。

学习者做练习时，应注意题目不但要求正确性，还要求效率。在编程平台上提交的程序运行通过时，反馈信息一般为 Accepted，本文将其简称为 AC。如果遇到个别题目因运行时间超时而不被 AC，就需要考虑选择 C++ 语言。不过用 Python 学习编程，有个特别大的好处。因为有的题目用 C++ 语言提交就直接 AC 了，但是用 Python 编写会超时，这时学生会琢磨如何优化，例如更换数据结构、改进算法等。这对于学习者理解相关原理、形成计算

思维特别有帮助。问题求解的关键在于算法，如果这个能力练好了，那么语言之间的可移植性、程序的模块化程度、甚至效率等方面的差异就容易解决了。

需要说明的是，本书不会在编程语言的语法知识上花费太多篇幅，因为市面上有大量的编程语言书籍可以参考，网上也有组织得很系统的参考手册。2.1和2.2节只是简单介绍一些概念并提供编程语言的学习指引。

## 2.1 Python基础

本节对 Python 重要的基本语法给出总结，包括基本数据类型，控制结构和函数。其他细节可以参考 <https://docs.python.org/3/tutorial/index.html>, <https://www.runoob.com/python3/python3-tutorial.html>, Goodrich 等人在2013年出版的《Data Structures and Algorithms in Python》 [9] 或者 Matthes 在2019年出版的《Python 编程：从入门到实践》 [10]。

Python的编程环境有多种，推荐 Anaconda (<https://www.anaconda.com>) 自带的 spyder集成开发环境 (Integrated development environment, IDE) ， spyder 启动方法详见 2.5.2 节；同学们经常用到的开发环境还有 PyCharm；另外，在线运行的程序使用 <https://leetcode-cn.com/playground/new/empty/> 也方便，它可以接收矩阵形式的输入数据。

为了熟悉和掌握基本语法，需要完成 30~40 个简单的实践题目，难度级别相当于于3.1节中给出的 Easy\_Level1 和 Easy\_Level2。学习 Python 语法，及之后进阶的学习，如果需要纸质书，可以参考附录3B中“有同学希望推荐纸质书”。

### 2.1.1 基本数据类型

在编程语言中，变量 (variable) 用于存储信息，以便被引用或操作。在 Python 中，每个变量都是一个对象 (object)，基于变量的数据类型 (data type, 也称为 类 class)，解释器会为变量分配内存。变量可以通过变量名访问，变量命名规定，必须是大小写英文，数字和下划线的组合，并且不能用数字开头。

Python 的基本数据类型如表2-1所示。如果一个类的每个对象在实例化时有一个固定的值 (immutable)，并且随后不能被改变，那么这个类就是不可变的。例如，float 类是不可变的。

表2-1 Python语言基本数据类型

Class	Description	Immutable
bool	Boolean value	✓
int	integer (arbitrary magnitude)	✓
float	floating-point number	✓
list	mutable sequence of objects	
tuple	immutable sequence of objects	✓
str	character string	✓
set	unordered set of distinct objects	
frozenset	immutable form of set class	✓
dict	Associate mapping (aka dictionary)	

把基本数据类型结构化表示出来，如图2-1所示。

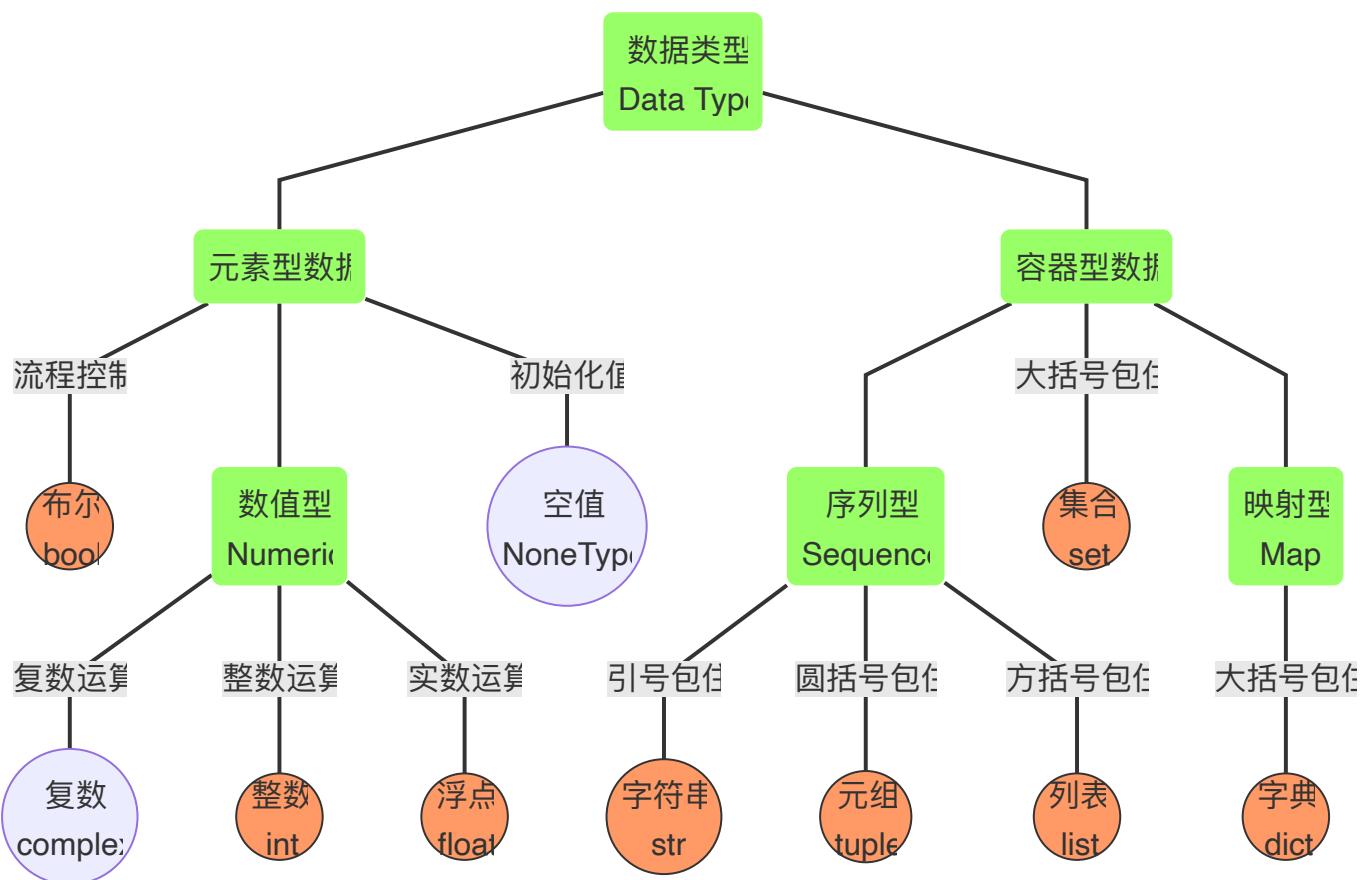


图2-1 Python语言基本数据类型

程序中离不了按自己需要的功能定义函数，函数（function，或者 method）就是一段代码，包括函数名和组成函数体的多行语句。在程序的其他部分可以通过函数名调用这段代码。Python 的每个类都提供了很多常用的函数，例如图2-2中的 list.append(x), list.sort() 分别是对列表进行元素追加和排序。对于 Python 的基本数据类型可以通过运算符（operator）或者函数进行操作，对基本数据类型的操作总结如图2-2所示。

Main data types		List operations	List methods
<code>boolean = True / False</code> <code>integer = 10</code> <code>float = 10.01</code> <code>string = "123abc"</code> <code>list = [ value1, value2, ... ]</code> <code>dictionary = { key1:value1, key2:value2, ... }</code>		<code>list = []</code> defines an empty list <code>list[i] = x</code> stores x with index i <code>list[i]</code> retrieves the item with index i <code>list[-1]</code> retrieves last item <code>list[i:j]</code> retrieves items in the range i to j <code>del list[i]</code> removes the item with index i	<code>list.append(x)</code> adds x to the end of the list <code>list.extend(L)</code> appends L to the end of the list <code>list.insert(i,x)</code> inserts x at i position <code>list.remove(x)</code> removes the first list item whose value is x <code>list.pop(i)</code> removes the item at position i and returns its value <code>list.clear()</code> removes all items from the list <code>list.index(x)</code> returns a list of values delimited by x <code>list.count(x)</code> returns a string with list values joined by S <code>list.sort()</code> sorts list items <code>list.reverse()</code> reverses list elements <code>list.copy()</code> returns a copy of the list
Numeric operators	Comparison operators	Dictionary operations	String methods
+ addition - subtraction * multiplication / division ** exponent % modulus // floor division	== equal != different > higher < lower >= higher or equal <= lower or equal	<code>dict = {}</code> defines an empty dictionary <code>dict[k] = x</code> stores x associated to key k <code>dict[k]</code> retrieves the item with key k <code>del dict[k]</code> removes the item with key k	<code>string.upper()</code> converts to uppercase <code>string.lower()</code> converts to lowercase <code>string.count(x)</code> counts how many times x appears <code>string.find(x)</code> position of the x first occurrence <code>string.replace(x,y)</code> replaces x for y <code>string.strip(x)</code> returns a list of values delimited by x <code>string.join(L)</code> returns a string with L values joined by string <code>string.format(x)</code> returns a string that includes formatted x
Boolean operators	Special characters	Dictionary methods	
and logical AND or logical OR not logical NOT	# coment \n new line \<char> scape char	<code>dict.keys()</code> returns a list of keys <code>dict.values()</code> returns a list of values <code>dict.items()</code> returns a list of pairs (key,value) <code>dict.get(k)</code> returns the value associated to the key k <code>dict.pop()</code> removes the item associated to the key and returns its value <code>dict.update(D)</code> adds keys-values (D) to dictionary <code>dict.clear()</code> removes all keys-values from the dictionary <code>dict.copy()</code> returns a copy of the dictionary	
String operations		Dictionary methods	
<code>string[i]</code> retrieves character at position i <code>string[-1]</code> retrieves last character <code>string[i:j]</code> retrieves characters in range i to j			

Legend: x,y stand for any kind of data values, s for a string, n for a number, L for a list where i,j are list indexes, D stands for a dictionary and k is a dictionary key.

图2-2 Python语言基本数据类型操作（注：图片来源为 sixthresearcher.com，2022年5月）

## 2.1.2 控制结构

通过操作符如加减乘除，把基本数据类型构造成表达式，再结合控制结构（Control structures），就可以实现结构化编程。控制结构包括条件分支（conditionals）和循环结构（loops），如图2-3, 2-4所示。

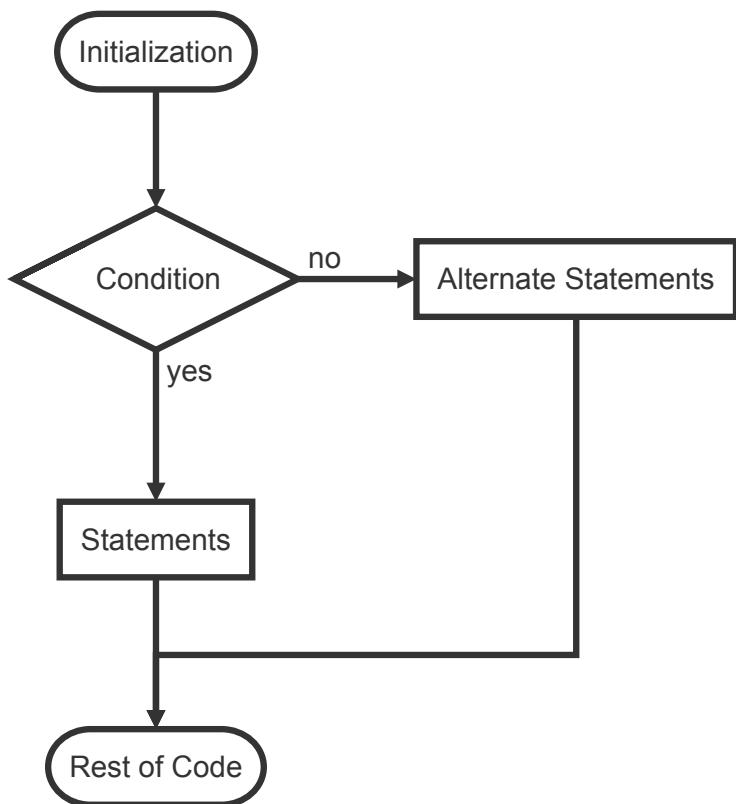


图2-3 条件分支控制结构

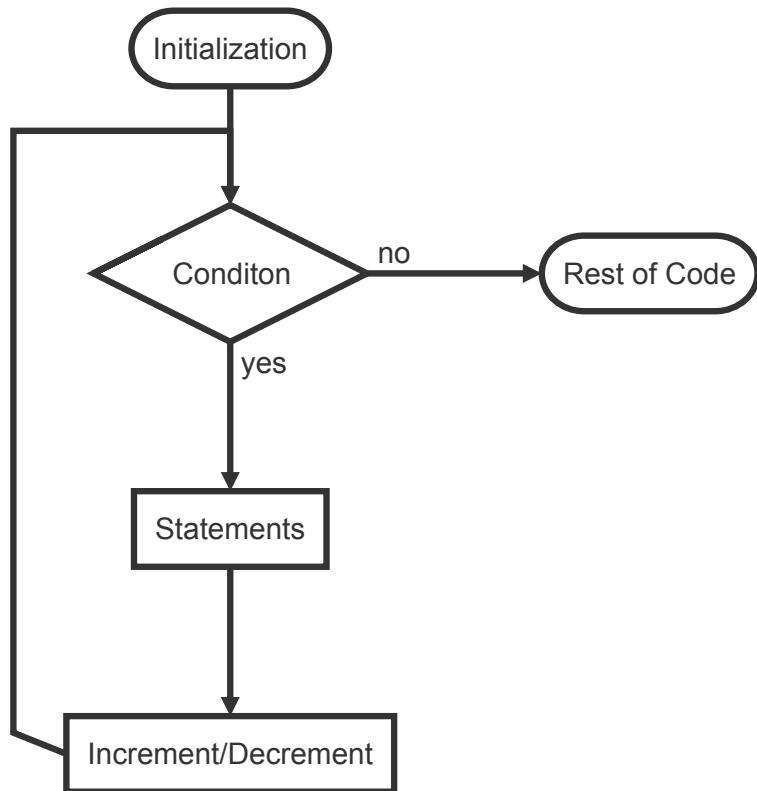


图2-4 循环控制结构

## 2.2 C++ 基础

当你遇到不会的题目时，通常在网上能搜索到的是C/C++代码，因此就需要大家能了解一些C/C++语言。另外，编程平台上有的题目对运行效率有要求，当用 Python 编写的代码超时不被接受时，可以尝试改写为 C++ 或者 C 语言的代码提交。由于 C 语言是 C++ 的子集，因此本节不单独介绍 C 语言。

1 C++ 是一种高级语言，它是由贝尔实验室 的 Bjarne Stroustrup 于 1979 年开始设计开发的。C++ 扩充和完善了 C 语言，是一种面向对象的程序设计语言。C++ 语言的学习与 Python 语言学习要求一样，读者可以先掌握基本语法，包括基本数据类型，控制结构和函数。其他细节请参考  
<https://www.cplusplus.com/doc/tutorial/>，或者 runoob.com 网站中 C++ 部分。

2  
3 C++ 程序可以使用 Leetcode 在线运行 (<https://leetcode-cn.com/playground/new/empty/>)。如果你使用的是 macOS 系统，可以用操作系统自带的 g++ 编译器在终端窗口中编译生成可执行文件，还可以使用 Xcode，这是一个运行在 Mac OS X 上的集成开发工具。如果你是在 Windows 系统中，可以使用 CodeBlocks 集成开发环境 (<https://www.codeblocks.org>)。

C++ 语言的基本数据类型有整型 (Integer)，字符型 (Character)，布尔型 (Boolean)，浮点型 (Floating Point)，双浮点型 (Double Floating Point)，无类型 (Void)，宽字符型 (Wide Character)。C++ 中可用的数据类型修饰器有 Signed, Unsigned, Short, Long。表2-4总结了内置数据类型与类型修饰符结合时的表示范围，其中

$$2^8 = 256, 2^{16} = 65,536, 2^{32} = 4,294,967,296, 2^{64} = 18,446,744,073,709,551,615.$$

表2-4 C++ 语言基本数据类型

数据类型	占用字节	可表示的数值范围
short int	2	$-2^{15}$ to $2^{15} - 1$
unsigned short int	2	0 to $2^{16} - 1$
unsigned int	4	0 to $2^{32} - 1$
int	4	$-2^{31}$ to $2^{31} - 1$
long int	4	$-2^{31}$ to $2^{31} - 1$
unsigned long int	4	0 to $2^{32} - 1$
long long int	8	$-2^{63}$ to $2^{63} - 1$
unsigned long long int	8	0 to $2^{64} - 1$
signed char	1	$-2^7$ to $2^7 - 1$
unsigned char	1	$-2^7$ to $2^7 - 1$
float	4	0 to $2^8 - 1$
double	8	
long double	16	

C++ 语言的基本数据类型及操作总结如图2-8, 2-9所示。

# Cheatography

## C++ Cheatsheet(inc. C++2011) Cheat Sheet by Leupi via cheatography.com/5907/cs/1197/

Standard Systemtypes and their range	
int8_t	-128 to 127
uint8_t	0 to 255
int16_t	-32768 to 32767
uint16_t	0 to 65535
int32_t	-2147483648 to 2147483647
uint32_t	0 to 4294967295
int64_t	-9.2 * 10 <sup>18</sup> to 9.2 * 10 <sup>18</sup>
uint64_t	0 to 1.8 * 10 <sup>19</sup>

Literals & Co.	
255	Integer
0xaf	Hexadecimal Integer
1234.0	double
234.243f	float
true	bool
"Hello World"	string/c-string
'a'	char

Bitwise Operators	
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Shift left
>>	Shift right

Boolean Logic	
==	Test of equality
!=	Test of non-equality
<	Less than
>	Greater than
<=	Less than or equal
>=	Greater than or equal
!	NOT
&&	AND
	OR

Pointers: Quick and dirty	
&	Gets RAM address of Variable(to save into pointer)
*	Dereferences pointer(returns it's content) or defines a variable to be a pointer
->	Access pointer class member. same as (*pointer).member()
new	Create new object on heap, returns pointer to object
delete	Remove object at the pointer on heap

Pointers allocate space on heap, normal variables on stack!

using replaces typedef	
//typedef is deprecated	typedef uint32_t uint32;

//now: using directive!

//using identifier = type\_name;  
using uint32 = uint32\_t;

图2-8 C++ 语言基本数据类型操作 (1) (注: 图片来源为 cheatography.com/leupi/, 2022年6月)

<b>Preprocessor</b>	Boolean expressions in C++ are evaluated left-to-right!	<b>Auto Datatype</b>
#include <LIB>	Includes standard header file LIB	//auto is an automatic datatype: int x = 4; //equals: auto y = 4; //works for most cases, esp. STL: for(std::vector<int>::iterator it = v.begin().....) //with auto: for(auto it = v.begin(); it != v.end(); ++it)
#include "Header.h"	Includes Header.h file	
#define PI	Defines Easy Macro 3.14159265359	
#define f(a,b)	More Complex Macro a+b	
#undef PI	Remove Definition	
#ifdef PI	Checks for definition, if true, code will be compiled	
#else	Else part of the above if- statement	
#endif	Ends the if-statement	
<b>Nullpointer</b>		
int* pInt; //old, deprecated: if(pInt == NULL) //new: if(pInt == nullptr) //still valid: if(!pInt)	+ addition - subtraction * multiplication / division % modulo ++var increase before evaluation var++ increase after evaluation condition ? result : alternative; short form of if-like structure :: Scope Operator -> pointer-to-member . Access member << >> Bitshift(with streams: input/output)	
		<b>Compile-time assertion check</b> static assert(bool_constexpr, string)
		<b>Multithreading</b> //Thread Creation & Management: void thread_func(int a) { std::cout << "My Number is: " << a; } main() { std::thread t1(thread_func(1)); t1.join();

图2-9 C++ 语言基本数据类型操作 (2/2) (注: 图片来源为 [cheatography.com/leupi/](http://cheatography.com/leupi/), 2022年6月)

编程语言的控制结构是相通的, C++ 的控制结构包括条件分支和循环结构, 与 Python 语言的一致。因此不再赘述。

## 2.3 把 C++ 程序翻译为 Python

当有题目做不出来时, 借助搜索引擎在网上通常可以找到经典题目的 C++ 程序解法, 但是可能没有现成的 Python 代码, 因此需要我们能看懂 C++ 程序, 并翻译到 Python 语言。我们来看两个把 C++ 翻译到 Python 的题目。考虑到版权问题, 本书不重复描述题面, 而是给出出处。请读者按照链接访问, 并对照本书给出的详细题解来思考。

下面两个举例的题目对于初学编程的同学来说比较难, 这也再次印证了翻译的必要性。因为简单题目同学自己就可以实现, 难的题目如果写不出Python程序, 可以参考C++程序, 然后翻译为Python程序。

### OJ04129: 变换的迷宫

bfs, <http://cs101.openjudge.cn/practice/04129>

这个题目容易想到广度优先搜索，但是它的数据搜索规模大、容易超时，因此需要用剪枝策略。由于每经过 k 单位时间，石头就会消失一次，那么当站在某点 (x,y) 时，时间为 t+k 和 t 时，之后行走面临的情境是完全一样的，那就意味着，对于某个状态的时间，可以取模后作为  $visited[x][y][time]$  的第三个变量，如果取模后的值代入发现已经访问过，那说明之前已经有更优越的情况出现过，不必再继续搜索了。思路参考如下，该程序是用 C++ 代码所写，原始出处为 <https://blog.csdn.net/dhc65376/article/details/101555903>

```
1  /* Bailian4129 变换的迷宫 */
2  #include <bits/stdc++.h>
3  using namespace std;
4  const int N = 100, K = 10;
5  char maze[N][N + 1];
6  bool vis[N][N][K];
7  struct Node {
8      int row, col, time;
9      Node(int r, int c, int t):row(r), col(c), time(t){}
10 };
11 int dr[] = {-1, 1, 0, 0};
12 int dc[] = {0, 0, -1, 1};
13 const int DL = sizeof(dr) / sizeof(int);
14 int main()
15 {
16     int t, r, c, k;
17     scanf("%d", &t);
18     while(t--) {
19         queue<Node> q;
20         scanf("%d%d%d", &r, &c, &k);
21         for(int i = 0; i < r; i++)
22             scanf("%s", maze[i]);
23         memset(vis, false, sizeof(vis));
24         int tr, tc, cnt = 0;
25         for(int i = 0; i < r; i++)
26             for(int j = 0; j < c; j++)
27                 if(maze[i][j] == 'S') {
28                     q.push(Node(i, j, 0));
29                     vis[i][j][0] = true;
30                     if(++cnt == 2) break;
31                 } else if(maze[i][j] == 'E') {
32                     tr = i;
33                     tc = j;
34                     if(++cnt == 2) break;
35                 }
36         while(!q.empty()) {
37             Node t = q.front();
38             if(t.row == tr && t.col == tc) break;
39             q.pop();
40             for(int i = 0; i < DL; i++) {
41                 int nrow = t.row + dr[i];
42                 int ncol = t.col + dc[i];
43                 if(nrow < 0 || nrow >= r || ncol < 0 || ncol >= c)
44                     continue;
45                 if(vis[nrow][ncol][(t.time + 1) % k])
```

```

46         continue;
47     if((t.time + 1) % k && maze[nrow][ncol] == '#') // 时间是K 的倍数时, 迷
宫中的石头就会消失
48         continue;
49     vis[nrow][ncol][(t.time + 1) % k] = true;
50     q.push(Node(nrow, ncol, t.time + 1));
51 }
52 }
53 if(q.empty())
54     printf("Oop!\n");
55 else
56     printf("%d\n", q.front().time);
57 }
58 return 0;
59 }
```

我们将上述代码翻译为 Python 语言, 请读者注意不同数据组之间的初始化。

```

1 arr2 = lambda m,n : [ [ ' ' for j in range(n)] for i in range(m) ]
2 arr3 = lambda m,n,l : [ [ [False for k in range(l)] for j in range(n)] for i in
range(m) ]
3
4 N = 100
5 K = 10
6
7 class Node:
8     def __init__(self, r=0, c=0, t=0):
9         self.row = r
10        self.col = c
11        self.time = t
12
13 dr = [-1, 1, 0, 0]
14 dc = [0, 0, -1, 1]
15
16 for _ in range(int(input())):
17     maze = arr2(N, N)          # 注意不同数据组之间的初始化
18     vis = arr3(N, N, K)
19     q = []
20     r,c,k = map(int, input().split())
21     for i in range(r):
22         maze[i][:c] = list(input())
23
24     tr = tc = cnt = 0;
25     for i in range(r):
26         for j in range(c):
27             if maze[i][j] == 'S':
28                 q.append(Node(i, j))
29                 vis[i][j][0] = True
30                 cnt += 1
31                 if cnt == 2: break
```

```

32         elif maze[i][j] == 'E':
33             tr = i
34             tc = j
35             cnt += 1
36             if cnt == 2: break
37
38     while(len(q)):
39         t = q[0] # t : Node
40         if t.row == tr and t.col == tc: break
41         q.pop(0)
42         for i in range(4):
43             nrow = t.row + dr[i]
44             ncol = t.col + dc[i]
45
46             if nrow < 0 or nrow >= r or ncol < 0 or ncol >= c:
47                 continue
48
49             # 剪枝很容易能知道, 由于每过k单位时间, 石头就会消失一次, 那么当我们站在某点 (x,y)
50             时,
51             # 时间为 t+k 和 t 时, 它们之后行走面临的情境是完全一样的, 那就意味着,
52             # 对于某个状态的时间, 我们可以取模后作为 visited[x][y][time] 的第三个变量,
53             # 如果取模后的值代入发现已经访问过, 那说明之前已经有更优越的情况出现过, 不必再继续搜索
54             了.
55
56             if vis[nrow][ncol][(t.time + 1) % k]:
57                 continue
58
59             # 时间是k 的倍数时, 迷宫中的石头就会消失
60             if (t.time + 1) % k and maze[nrow][ncol] == '#':
61                 continue;
62             vis[nrow][ncol][(t.time + 1) % k] = True
63             q.append(Node(nrow, ncol, t.time + 1))
64
65     if len(q) == 0:
66         print("Oop!")
67     else:
68         print(q[0].time)

```

## 1427B. 国际象棋作弊器 Chess Cheater

greedy/implementation/sortings, 1400, <https://codeforces.com/problemset/problem/1427/B>

思路: <https://www.bbsmax.com/A/WpdKEVpmJV/>

请注意, 分数等于  $[score = 2 \cdot \# \{wins\} - \# \{winning\_streaks\}]$ , 连胜是连续获胜的最大顺序。

在下面的说明中, 变量( $\# \{wins\}$ ,  $\# \{winning\_streaks\}$ )始终与初始情况相关。

如果  $(k + \# \{wins\}) \geq n$ , 则有可能赢得所有比赛, 因此答案为  $(2n-1)$ 。

否则，很明显，要转换 k 获胜中的 k 损失。因此，作弊后，获胜次数将为( $k + \# \{wins\}$ )。考虑到以上公式，仍然仅是要减少获胜间隔的数量。

如何才能减少获胜间隔的数量？非常直观的是，将从长度最短的差距开始，以“填补”连续的获胜间隔之间的差距。可以证明，如果没有填补 g 个缺口（即在作弊之后，g 个缺口仍然至少包含一个损失），则至少有 $g + 1$  个获胜间隔。

实现过程如下。通过线性扫描，可以找到间隙的长度，然后对它们进行排序。最后，计算可以选择的总和 ( $\leq k$ ) 的数量。答案是

[2·  $(k + \# \{wins\}) - \# \{winning\_streaks\} + \# \{gaps\_we\_can\_fill\}$ ]

解决方案的复杂度为  $O(\log(n))$ 。

```
1 #include<bits/stdc++.h>
2 #define ms(a,b) memset(a,b);
3 using namespace std;
4 typedef long long ll;
5 int main() {
6     //freopen("in.txt", "r", stdin);
7     ios_base::sync_with_stdio(false), cin.tie(0), cout.tie(0);
8     int T;
9     cin >> T;
10    for (int t = 1; t <= T; t++) {
11        int N, K;
12        cin >> N >> K;
13        string S;
14        cin >> S;
15        int winning_streaks_cnt = 0;
16        int wins = 0;
17        int losses = 0;
18        vector<int> losing_streaks;
19        for (int i = 0; i < N; i++) {
20            if (S[i] == 'W') {
21                wins++;
22                if (i == 0 || S[i - 1] == 'L') winning_streaks_cnt++;
23            }
24            if (S[i] == 'L') {
25                losses++;
26                if (i == 0 || S[i - 1] == 'W') losing_streaks.push_back(0);
27                losing_streaks.back()++;
28            }
29        }
30        if (K >= losses) {
31            cout << 2 * N - 1 << "\n";
32            continue;
33        }
34        if (wins == 0) {
35            if (K == 0) cout << 0 << "\n";
36            else cout << 2 * K - 1 << "\n";
37            continue;
38        }
39    }
40 }
```

```

38     }
39     if (S[0] == 'L') losing_streaks[0] = 1e8;
40     if (S[N - 1] == 'L') losing_streaks.back() = 1e8;
41     sort(losing_streaks.begin(), losing_streaks.end());
42     wins += K;
43     for (int ls : losing_streaks) {
44         if (ls > K) break;
45         K -= ls;
46         winning_streaks_cnt--;
47     }
48     cout << 2 * wins - winning_streaks_cnt << "\n";
49 }
50 }
```

我们将上述代码翻译为 Python 语言。

```

1 # ref: https://www.bbsmax.com/A/WpdKEVpmJV/
2 for _ in range(int(input())):
3     N,K = map(int, input().split())
4     S = input()
5     winning_streaks_cnt = wins = losses = 0
6     losing_streaks = []
7
8     for i in range(N):
9         if S[i] == 'W':
10             wins += 1
11             if i==0 or S[i-1]=='L':
12                 winning_streaks_cnt += 1
13
14         if S[i]=='L':
15             losses += 1
16             if i==0 or S[i-1]=='W':
17                 losing_streaks.append(0)
18                 losing_streaks[-1] = losing_streaks[-1] + 1
19
20     if K >= losses:
21         print(2*N-1)
22         continue
23
24     if wins == 0:
25         if K == 0:
26             print(0)
27         else:
28             print(2*K-1)
29             continue
30
31     if S[0]=='L':
32         losing_streaks[0] = 1e8
33     if S[-1]=='L':
34         losing_streaks[-1] = 1e8
```

```

35
36     losing_steaks.sort()
37     wins += K
38     for ls in losing_steaks:
39         if ls > K:
40             break
41
42         K -= ls
43         winning_steaks_cnt -= 1
44
45     print(2*wins - winning_steaks_cnt)

```

## 2.4 终端窗口命令

有时候需要在命令行方式执行或者调试程序，因此需要掌握终端窗口命令。两种常用操作系统是 macOS 和 Windows。终端窗口的开启，在 Windows 中是打开 cmd 窗口，在 macOS 中是打开 terminal 窗口。

### 2.4.1 Windows 中的 cmd 窗口

在 Windows 系统中，点击任务栏左下角窗口图标“开始”按钮；在弹窗中输入“cmd”，点击“打开”；则“命令提示符”应用的黑窗口就打开了，如图2-10所示。这个窗口，称为 cmd 窗口或者 dos 窗口。在 cmd 窗口中运行 help 指令，可以列出常用命令，如表2-5所示。



图2-10 Windows 的 cmd 窗口

表2-5 常用 cmd 窗口命令

命令	描述
CD	显示当前目录的名称或将其更改
CHDIR	显示当前目录的名称或将其更改
CLS	清除屏幕
COMP	比较两个或两套文件的内容
COPY	将至少一个文件复制到另一个位置
DIR	显示一个目录中的文件和子目录
EXIT	退出 CMD.EXE 程序（命令解释程序）
HELP	提供 Windows 命令的帮助信息
MD	创建一个目录
MKDIR	创建一个目录
MORE	逐屏显示输出
MOVE	将一个或多个文件从一个目录移动到另一个目录
PATH	为可执行文件显示或设置搜索路径
REN	重命名文件
RENAME	重命名文件
SET	显示、设置或删除 Windows 环境变量

到cmd窗口。dir看看文件是否都在。

## 2.4.2 macOS 中的 terminal 终端

在 macOS 系统中，开启终端窗口的方法是：点击 Dock 中的 Launchpad 图标 ，在搜索栏中输入“terminal”，然后点击 Terminal，如图2-11所示。常用 terminal 命令，如表2-6所示。

```
Last login: Thu Jun 16 12:52:51 on ttys001
The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
(base) MBP15deMacBook-Pro:~ MBP15$
```

图2-11 macOS 的 terminal 窗口

表2-6 常用 Terminal命令

命令	描述
cd	进入到某个文件路径下
cd ~/Desktop/	进入桌面位置
pwd	当前文件路径
cd ..	返回上一级目录
find *.py	查找当前目录下所有的 py 文件
cd -	返回上一个访问的目录
rmdir	删除空目录
mv dir1 dir2	移动或重命名一个目录
ctrl + c	终止
ls	查看当前目录下的文件夹和目录
mkdir	新建文件夹
touch	新建文件
cp	拷贝文件
clear	清除屏幕
file	显示文件类型

## 2.5 调试代码

写出来的程序通常要经过多次调试（debug），不会一次提交就被编程平台 AC。当问题稍微复杂时，代码也会变长一些，比如超过 20 行，可以用 print 输出变量状态信息，对 Python 代码的进行调试；还可以采用下面几种方法，本小节将依次直观的可视化调试工具 Pythontutor，Spyder 集成环境自带的调试工具，以及用题目的测试数据来调试的方法。

## 2.5.1 Pythontutor可视化

Pythontutor (<https://pythontutor.com>) ,是在线的代码执行可视化工具。它可以 Next 前进执行程序，也可以 Prev 后退执行，帮助我们了解计算机运行每一行代码时会发生什么。即对于抽象的计算机原理，如内存结构，能够直观的呈现出来，如图2-12所示。它简单易用，点击“Start writing and visualizing code now”开始使用即可，但是它的缺点是不能调试长代码及迭代次数多的代码。